A PROOF OF PIERI'S FORMULA USING GENERALIZED SCHENSTED INSERTION ALGORITHM FOR RC-GRAPHS.

MIKHAIL KOGAN AND ABHINAV KUMAR

ABSTRACT. We provide a generalization of the Schensted insertion algorithm for rc-graphs of Bergeron an Billey [1]. The new algorithm is used to give a new proof of Pieri's formula.

1. INTRODUCTION

RC-graphs were first defined by Fomin and Kirillov [3] and later studied by Bergeron and Billey [1]. These are a set of objects that encode the monomials contributing to the expansion of Schubert polynomials. These polynomilas were introduced by Lascoux and Schutzenberger [6] [7] and described at length by Macdonald [8] and by Manivel [9].

A central problem in the theory of Schubert polynomials is to provide effective ways of computing the generalized Littlewood-Richardson coefficients c_{vw}^u in the expansion

$$(1) \qquad P_v P_w = \sum_u c_{vw}^u P_u$$

where P_w is the Schubert polynomial of a permutation w. (We have used the fact that Schubert polynomials form a basis of all polynomials to get a unique such expansion.)

The first attempt to compute the generalized Littlewood-Richardson coefficients using rc-graphs was made in [1], where Monk's formula was proved using a generalized Schensted insertion algorithm. Later, in [5], this generalized algorithm was used to compute a more general set of Littlewood-Richardson coefficients. Unfortunately, this algorithm does not work in general and, in particular, fails to give a proof Pieri's formula. In this paper we will modify this algorithm to inserting a whole row of elements at once instead of an element by element insertion. This generalization will be used to prove Pieri's formula.

Let us give a statement of Pieri's formula, whose different modifications have been conjectured in [6], [1] and later proved in [11], [10], [13].

Theorem 4.1.
$$P_w P_{\sigma[r,m]} = \sum P_{w'}$$
where the sum is over all $w' = w t_{a_1,b_1} ... t_{a_m,b_m}$ such that the b_i's are distinct and greater than r, $a_1 \le a_2 \le ... \le a_m \le r$, satisfying $l(w') = l(w) + m$ and $w'(b_i) < w'(b_j) < w'(a_i)$ for every $i < j$ with $a_i = a_j$.

In the above statement, $\sigma[r,m]$ stands for the permutation $[1, 2, ..., r-1, r+m, r+1, r+2, ...]$. Note that $P_{\sigma[r,m]}$ is equal to the homogeneous symmetric polynomial $h_m(x_1, ..., x_r)$.

We will define Schubert polynomials, rc-graphs and point out their relationship in Section 2. The generalized insertion algorithm will be given in Section 3, while in Section 4 we will prove Pieri's formula by providing the inverse algorithm.

2. RC-GRAPHS

Let us decide on some notation and state some basic facts about permutations and Schubert polynomials first. For a permutation $w \in S_n$, the symmetric group on n elements, we can write w as $[w_1, ..., w_n]$, abbreviating $w(i)$ to w_i. We let t_{ab} denote the transposition which takes a to b and vice versa, leaving other elements fixed, and write s_i for $t_{i,i+1}$. It can be shown that $s_1, ..., s_{n-1}$ generate S_n with the relations

$$
\begin{aligned}
s_i^2 &= 1, \\
s_i s_j &= s_j s_i \text{ if } |i - j| \geq 2, \\
s_i s_{i+1} s_i &= s_{i+1} s_i s_{i+1}.
\end{aligned}
$$

For a permutation $w \in S_n$, we let the length $l(w)$ of w be the number of inversions of w, i.e. the number of pairs (i, j) with $i < j$ and $w(i) > w(j)$. A string $a_1 a_2 ... a_p$ such that $s_{a_1} ... s_{a_p} = w$ with p minimal is called a shortest word for w. It can be shown that this p is just the length of w, and any two shortest word decompositions can be transformed into each other using the last two relations given above, i.e. that the graph of reduced words is transitive under the corresponding operations on the strings. Denote by $R(w)$ the set of all the shortest words for w.

This enables us to define Schubert polynomials. Let ∂_i be an operator which acting on a polynomial $f(x_1, ..., x_n)$ to the left, yields the polynomial

$$
\frac{f(x_1, ..., x_i, x_{i+1}, ..., x_n) - f(x_1, ..., x_{i+1}, x_i, ...x_n)}{x_i - x_{i+1}}
$$

It can be verified that

$$
\begin{aligned}
\partial_i^2 &= 0, \\
\partial_i \partial_j &= \partial_j \partial_i \text{ if } |i - j| \geq 2 \\
\partial_i \partial_{i+1} \partial_i &= \partial_{i+1} \partial_i \partial_{i+1}
\end{aligned}
$$

Thus for any permutation w, we may define ∂_w as $\partial_{a_1} ... \partial_{a_p}$, where $a_1 ... a_p$ is *any* reduced word for w. Now the Schubert polynomial P_w is defined to be

$$
P_w = \partial_{w^{-1} w_0} x_1^{n-1} x_2^{n-2} ... x_{n-1}^1 x_n^0
$$

where $w_0 = [n, ..., 1]$ is the longest word in S_n.

Now we come to rc-graphs. Given a reduced word $a_1 ... a_p$ for w, we say that a sequence $\alpha_1, ..., \alpha_p$ is compatible for the word if

$$
\begin{aligned}
\alpha_1 \leq \alpha_2 &\leq .... \leq \alpha_p \\
\alpha_i &\leq a_i, \forall\, i \\
\alpha_i < \alpha_{i+1} &\text{ if } a_i < a_{i+1}
\end{aligned}
$$

Denote by $C(\mathbf{a})$ the set of all compatible sequences for $\mathbf{a} \in R(w)$.

Given such a compatible sequence we make an rc-graph by placing intersections (see Figure 1) at all the positions $(\alpha_k, a_k - \alpha_k + 1)$, these form a subset of $\{1, 2, ...\} \times \{1, 2, ...\}$. Now we draw strands starting at each row and winding their way up. Wherever there is an intersection symbol, we make the two strands there intersect. It is easy to show by induction that the intersections represent exactly the transpositions in the decomposition of w and that the strand starting at the ith

row ends up at the w_i^{th} column. Also, it follows from this that no two strands intersect twice, otherwise we could remove the 2 intersections and get exactly the same permutation, since any other pair of strands that intersected before still intersect. Then the word which led to the graph would not be reduced, a contradiction. Conversely, given a graph with intersections such that no two strands intersect twice, this immediately gives us a reduced word for the permutation and also a compatible sequence (we look at the intersections in order of increasing row number, going from right to left in each row, the row number gives the α_i and to get a_i we add the row number and column number and subtract 1.) For more details see [1].
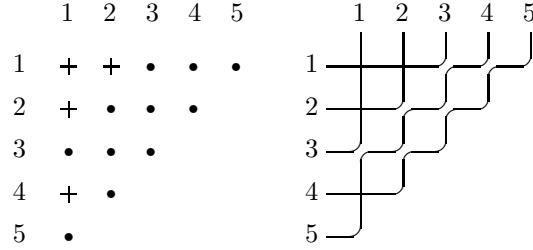

Figure 1: Two ways of drawing an rc-graph.

Denote by $\mathcal{RC}(w)$ the set of all rc-graphs, which correspond to w. For $D \in \mathcal{RC}(w)$ let $x^D = \prod_{(\alpha_k, a_k - \alpha_k + 1) \in D} x_{\alpha_k}$. Now we state a theorem proved in [2] and in [4].

Theorem 2.1. *For any permutation $w \in S_\infty$,*

$$P_w = \sum_{\mathbf{a} \in R(w)} \sum_{\alpha_1 \ldots \alpha_p \in C(\mathbf{a})} x_{\alpha_1} \ldots x_{\alpha_p} = \sum_{D \in \mathcal{RC}(w)} x^D$$

On an rc-graph we can define a ladder move (of size m) as shown in the figure. Note that after the move we get an rc-graph corresponding to the same permutation (since the minimal number of intersections is preserved and the same strands intersect in the two figures).
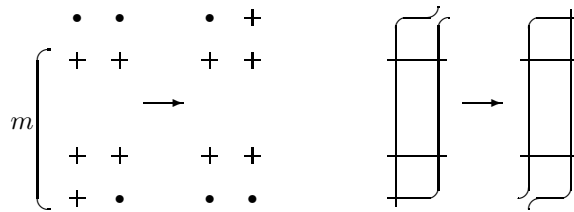

Figure 2: Ladder moves

We shall call the inverse operation an inverse ladder move. Billey and Bergeron in [1] prove that all graphs in $\mathcal{RC}(w)$ can be obtained by applying a succession of ladder moves to $D_{bot}(w) = \{(i, c) : c \leq m_i\}$ (where $m_i = \#\{j : j > i \text{ and } w_j > w_i\}$), which is the "bottom" rc-graph corresponding to w. Note that the bottom graph has the property that the rows are left-justified. For instance, the graph shown in Figure 1 is a bottom graph.

3. Generalizing insertion algorithm.

Recall that Pieri's rule seeks to give a formula for multiplication of a Schubert polynomial with $P_{\sigma[r,m]}$ (a homogeneous symmetric polynomial). In other words, it computes the generalized Littlewood-Richardson coefficients c_{vw}^u for $w = \sigma[r,m]$, where the coefficients are define by

$$P_v P_w = \sum_u c_{vw}^u P_u.$$

To enable us to find this formula, we need a modification of the insertion algorithm, which was defined in [1] and used to compute some set of Littlewood-Richardson coefficients in [5]

Notice that each rc-graph for the permutation $\sigma[r,m]$ can be given by specifying the number of intersections in the rows on or above row r. Assume this rc-graph is given by k_s elements in the row i_s, where $i_1 > i_2 > ... > i_t$ and $\sum k_s = m$. Below is the algorithm which inserts this rc-graph into a general rc-graph.

Algorithm: Given an rc-graph D, and a level r, suppose we have to insert k_s elements into row i_s, where $i_1 > i_2 > ... > i_t$ into D ($r \geq i_1$). We keep an ordered sequence of (a_i, b_i)'s during the course of the algorithm, such that $a_1 \leq a_2 \leq ... \leq a_l$ and all b_i's are distinct. Initially the sequence is empty.

Proceed row by row as follows. Starting with row i_1, find the rightmost position (i_1, j) where the configuration is one of the following, as shown in Figure 3:

$$s \underline{}\Big\rfloor\underline{} \qquad \begin{array}{l} \text{with } s \leq r < t, \\ t \neq b_j \text{ for any } j. \end{array} \qquad\qquad b_i \underline{}\Big\rfloor\underline{} \qquad \begin{array}{l} \text{with } r < t, \\ t \neq b_j \text{ for any } j. \end{array}$$
$$\quad t \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad t$$

Figure 3.

Add this intersection. If we are in the first case, to the list of (a,b)'s add (s,t). To preserve the order of a_i's, if s is equal to some $a_i = ... = a_{i+k}$, place (s,t) at the $i+k+1$ place. If we are in the second case add (a_i, t) just before where (a_i, b_i) is in the sequence. Proceed to do the same for the other intersections on the row i_1. Now we have a sequence of strands $a_1, b_1, ..., a_{k_1}, b_{k_1}$ where we know for sure that the b_i's are distinct and a_i's are ordered. Also, the graph we have obtained might not be an rc-graph. So we go to the next row and perform the following rectification procedure. Starting from the left, we look for either of the two configurations:

$$b_i \underline{}\Big| \qquad\qquad\qquad\qquad b_i \underline{}\Big| \qquad \text{with } a_i = a_j,$$
$$\qquad a_i \qquad\qquad\qquad\qquad\qquad\qquad b_j$$

Figure 4.

We remove the intersection, delete the pair (a_i, b_i) from the sequence of (a,b)'s. If the intersection was at position (i,j), find the maximal $j' < j$ such that the configuration at (i,j') is of the form shown in Figure 3. (Such a position must be encountered since the b_i we removed is greater than r and no longer a b_i, also the row number is less than r.) We add the intersection and to the list of (a,b)'s, we

add this new pair the same way it was described before. Then we look to the right
to see if there is another intersection of the form given in Figure 4, and rectify
that, and so on. When we are done with rectifying the intersections, we add what
intersections we must to this row in the same manner as for the previous row,
giving us some new pairs (a_j, b_j). We then proceed to the next row and repeat the
procedure. It is clear from the construction that the b's remain distinct and a's
remain ordered. $\qquad\square$

The rest of this section is concerned with proving the following:

Theorem 3.1. *The above algorithm produces an rc-graph.*

We show this by induction: we'll show that after dealing with a row ℓ (rectifying
it and inserting elements), the part of the graph below and including that row is
an rc-graph (i.e. no two elements intersect twice). Assume the this is true for the
row $\ell + 1$.

We will need a couple of lemmas and another algorithm first.

Lemma 3.2. *Assume we are the moment of the algorithm when we rectifying row
ℓ of the graph. Assume that at this moment $a = a_i = ... = a_{i+k}$. Then at the
row ℓ, the strands $b_i, ..., b_{i+k}$ are to the left to the strand a and they are ordered in
this row, the same way they are orderen in the (a, b) sequence. (See Figure 5 for
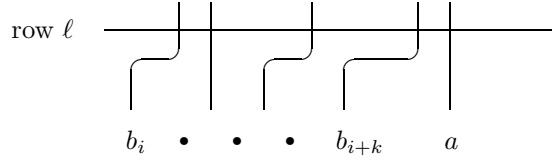illustration.)*

Figure 5. The order of b_i's strands at the row ℓ is the same as in the (a, b) sequence.

Proof: This is a trivial proof by induction. Clearly the order of starnds a
and $b_i, ..., b_{i+k}$ in the row ℓ can change only when any of them intersect, but such
intersections are shown on Figure 4 and they are removed during the algorithm. At
the same time, whem an intersection is added the addition of (a_j, b_j) to the list is
consistent with order of the starnds a and $b_i, ..., b_{i+k}$ in the row ℓ. $\qquad\square$

Lemma 3.3. *The permutation carried by the strands of the graph at each moment
of the algorithm is $wt_{a_1 b_1}...t_{a_{m'} b_{m'}}$ where $< (a_1, b_1), ..., (a_{m'}, b_{m'}) >$ is the sequence
upto that point in the algorithm.*

Proof: We again use induction: we need to check the above statement remains
true when we insert an intersection or delete one.

If we are in the first case of Figure 3, that is we are adding an intersection and
(s, t) to the list of (a, b)'s. We can easily see that adding this intersection multiplies
the permutation of the graph by $t_{s,t}$ on the right. So, if $< (a_1, b_1), ..., (a_{m'}, b_{m'}) >$
is the list before this insertion, the permutation of the graph after the insertion is
$wt_{a_1 b_1}...t_{a_{m'} b_{m'}} t_{s,t}$. Using the fact that t_{ab} and t_{cd} commute when a, b, c, d are all
distinct, we can move $t_{s,t}$ trough $t_{a_1 b_1}...t_{a_{m'} b_{m'}}$ to the left until some $a_j \le s$. But
this is exactly how the algorithm changes the list of (a, b)'s.

If we are in the second situation of Figure 3, then we multiply the permutation
$wt_{a_1 b_1}...t_{a_{m'} b_{m'}}$ by $t_{b_i,t}$ on the right. Then $t_{b_i,t}$ can be moved to the left until the

place i, since all b_j's are distinct and thus $t_{b_i,t}$ commutes with all t_{a_j,b_j} if $j > i$. After that we use the following identity, which shows why we have to add (a_i, t) to the (a, b) list on the ith place:

$$t_{a,b_i} t_{b_i,j} = t_{a,j} t_{a,b_i}.$$

Finally for the deletion, we use almost identical arguments in both cases of Figure 4, since the deletion of an intersection also multiplies the permutation of the graph by appropriate transposition. The only difference is in the second case, when we have to use the following trivial identity:

$$t_{a,b_i} t_{a,b_j} t_{b_i,b_j} = t_{a,b_j}$$

This concludes the proof of the Lemma.   □

To give a proof of Theorem 3.1 we have to introduce another algorithm (call it algorithm 2) that we perform on the intermediate graph. This algorithm will take the graph, which was rectified upto the row ℓ, and remove some intersections to produce an rc-graph of the original permutation w.

Algorithm 2. We start with the list of (a_i, b_i)'s and starting with the last row ℓ we have finished with, go down one row by one and from right to left in each row. We look for intersections of the form

Figure 6.

Whenever we see these, we remove the intersection and (a_i, b_i) from its place in the list. Note that both statements of Lemma 3.2 and Lemma 3.3 hold at each moment of this algorithm. Ultimately all the (a_i, b_i) will be removed from the list because each strand a_i definitely intersects the strand b_i at each moment of algorithm 2 by Lemma 3.2, so all of them will get removed and at the end we will get a graph and an empty list of (a, b)'s.   □

Now the result of algorithm 2 will be a graph with the permutation w by Lemma 3.3, with exactly $l(w)$ intersections. Therefore it will be an rc-graph for w. We will use this fact to prove that the main algorithm generates an rc-graph.

Now, we come back to the proof of Theorem 3.1. Suppose we have rectified some row and inserted some elements in it, and we want to show there are no double intersections below and including that row. Because of the inductive hypothesis, this works for the previous row $\ell + 1$ (base case for the first row is clear). Hence we only need to check that we have no double intersections in the row ℓ, after thiw has been rectified.

We will show that the only double intersection which might be introduced in the row ℓ by insertions in the lower rows will get removed by the algorithm and no new double intersections will be introduced in this row or any of the lower rows.

Let us show that after rectifying row $\ell+1$, and deleting all the intersections from figure 4 in the row ℓ at once, there are no strands, which intersect twice below row ℓ. For this apply Algorithm 2 to the graph constructed after rectifying row $\ell + 1$. Then the resulting graph is an rc-graph D of the permutation w. Start applying the inverse of Algorithm 2 to this rc-graph D, that is stard adding intesections to

D in the opposite order they were removed. It is then not difficult to see that since we are only adding intersections from Figure 6 and at each moment there are no double intersections below row $\ell+1$, the only double intersections at the row ℓ have to look like the ones on Figure 4. This shows that removing intersections from row ℓ we remove all the double intersection below this row.

Now let's check that after adding the intersections in the row ℓ we do not get new double intersections. Clearly the only way we could have done this is if we inserted an intersection in the position as in the second case of Figure 3, so that the strands b_i and t intersected before, in a lower row. (In the first case of Figure 3, the two strands could not have intersected in a lower row, since $s < t$.) So, we shall show that in the first case the strands b_i and t cannot intersect in a lower row, that is $b_i < t$. Proving this will finish the proof of Theorem 3.1.

After the row ℓ is rectified, we again apply Algorithm 2. Notice that this algorithm keeps moving the strand b_i to the left in the row ℓ till the last point, where it is removed off the list of (a_i, b_i)'s, at which point it moves right, even to the right of its *original* position. For example, in Figure 7, the circled intersections are opened during Algorithm 2. When the first one is opened, b_j is removed from the list and b_i moves from x to the left, to z, and when the second intersection is opened, b_i is removed and it moves to the right of x, to y.

Figure 7. During Algorithm 2, b_i moved from x to z and then to y in the row ℓ.

Therefore the position x of the strand b_i at the row ℓ at the start of Algorithm 2 is to the left of the position y at the row ℓ at the end of it.

Now we look at what happens after the b_i is moved off the list in Algorithm 2, possibly by some b_{i_1} (first it moved from x to z and then from z to y in the row ℓ). We look next at the history of b_{i_2} till it is moved of the list and so on, getting a figure like the following.

Figure 8. No strands between a and b_i can pass in the shown region between z and y.

Note that no two consecutive strands shown in the figure can intersect above the encircled spot (where one of them is moved off the list), because that intersection would then have to be removed earlier than the encircled one, by construction of the algorithm 2. Therefore, from the fact that the outcome of algorithm 2 is an rc-graph, we conclude that in the region shown, there can pass no strand between a and b_i (otherwise some double intersections must occur). Hence the t from the second case of Figure 3 must always be greater than b_i. This concludes the proof of Theorem 3.1.

Remark 3.4. Assume that the permutation w satisfies the following condition $w(i+1) > w(i)$ if $i > r$. Each rc-graph of such a permutatios has the following defining property: no two strand, which start below row r intersect. In this case it can be shown that the above algorithm is equivalent to one by one insertion. This is consistent with the results in [5], where the Pieri formula in this special case was proved using the one by one insertion. The above statement and the fact that one by one insertion does not work in general indicate that this algorithm provides the proper generalization of Bergeron-Billey algorithm of [1].

4. The proof of Pieri's formula.

To prove Pieri's formula using the above insertion algorithm, we will construct an inverse algorithm, which takes an rc-graph for $w' = wt_{a_1,b_1}...t_{a_m,b_m}$, such that the b_i's are distinct and greater than r, $a_1 \le a_2 \le ... \le a_m \le r$, satisfying $l(w') = l(w) + m$ and $w'(b_i) < w'(b_j) < w'(a_i)$ for every $i < j$ such that $a_i = a_j$, and produces an rc-graph for w and an rc-graph for $\sigma[r,n]$ (remember, that rc-graphs of $\sigma[r,m]$ are given by specifying the number of intersections k_s in in each row $s \le r$, such that $\sum k_s = m$). Given the insertion algorithm and its inverse, we will clearly produce a bijection:

$$\mathcal{RC}(w) \times \{k_1, ..., k_r : \sum k_i = m\} = \mathcal{RC}(w) \times \mathcal{RC}(\sigma[r,n]) \to \bigcup \mathcal{R}(w')$$

where w' ranges as stated above.

Below we will produce this inverse algorithm, which will automatically prove the following

Theorem 4.1.

$$P_w P_{\sigma[r,m]} = \sum P_{w'}$$

where the sum is over all $w' = wt_{a_1,b_1}...t_{a_m,b_m}$ such that the b_i's are distinct and greater than r, $a_1 \le a_2 \le ... \le a_m \le r$, satisfying $l(w') = l(w) + m$ and $w'(b_i) < w'(b_j) < w'(a_i)$ for every $i < j$ with $a_i = a_j$.

Inverse Algorithm.

We are given an rc-graph R' for a permutation $w' = wt_{a_1 b_1}...t_{a_m b_m}$ with the distinct b_i's, which are greater than r and with $a_1 \le a_2... \le a_m \le r$, $l(w') = l(w) + m$, and $w'(a) > w'(b) > w'(c)$ for all a, b, c such that (a, b) is some (a_s, b_s) and (a, c) is some (a_t, b_t) with $s > t$. The algorithm is defined as follows: starting from the top (first) row of the rc-graph, we look from right to left for the occurrence of one of the configurations shown below.
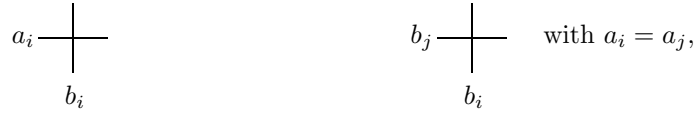
$$a_i \dashrightarrow \qquad b_j \dashrightarrow \qquad \text{with } a_i = a_j,$$
$$b_i \qquad\qquad\qquad b_i$$

Figure 9.

We then remove this intersection from the rc-graph and consequently remove (a_i, b_i) from the list of (a, b)'s. Immediately after we remove such an intersection we look to its right for a configuration of the form

$$t \longrightarrow \quad \begin{array}{l} \text{with } s \leq r < t, \\ t \neq b_i \text{ for any } i. \end{array} \qquad t \longrightarrow \quad \begin{array}{l} \text{with } r < t, \\ t \neq b_j \text{ for any } j. \end{array}$$
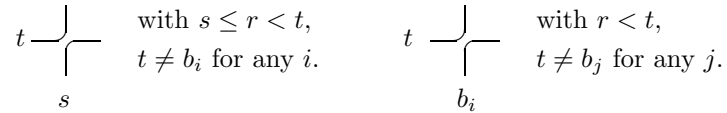$$s \qquad\qquad\qquad\qquad\qquad b_i$$

Figure 10.

If we don't find such a configuration, we move on with the algorithm, looking left along that row (and after that, going to the next row) for an intersection of the form shown. (At the same time we record the number of this row, as the numbers of these rows will give an rc-graph for $\sigma[r, m]$ at the end of the algorithm.) If we find such a configuration, we add the intersection at this place (i.e. the first configuration of this sort, looking to the right of the removed intersection) and add (r, s) or (a_i, t) to the list of (a, b)'s the same way it was done in the forward insertion algorithm, i.e we add (r, s) to be (a_i, b_i) to preserve the order of a's ($a_{i-1} \leq a_i < a_{i+1}$) and we add (a_i, t) right after (a_j, b_j). Then we proceed with the algorithm, looking left for more intersections to remove. We do this till the list of (a, b)'s becomes empty. At the end we are left with a graph for the permuation w. Moreover, the previously recorded rows, where intersections were removed from the rc-graph, produce an rc-graph for $\sigma[r, m]$.

The fact that he resulting graph is an rc-graph can be shown using the same sort of technique used for the forward algorithm. It is also evident that the algorithm is the inverse of the forward algorithm. This finishes the proof of Theorem 4.1.

References

[1] N. Bergeron and S. Billey, *RC-graphs and Schubert polynomials*, Experimental Math., **2** (1993), 257-269.

[2] S. Billey, W. Jockusch, R. Stanley. *Some combinatorial properties of Schubert polynomials.* J. Algebraic Combin. 2 (1993), no. 4, 345–374.

[3] S. Fomin and A. N. Kirillov, *Yang-Baxter equation, symmetric functions, and Schubert polynomials*, Discrete Mathematics **153**, (1996) 123-143.

[4] S. Fomin and R. Stanley, *Schubert polynomials and the nil-Coxeter algebra.* Adv. Math. 103 (1994), no. 2, 196–207.

[5] M. Kogan *RC-graphs and Littlewood-Richardson Rule.* preprint

[6] A.Lascoux and M. P. Schutzenberger, *Polynomes de Schubert*, C.R.Acad. Sci. Paris **294** (1982) 447-450.

[7] A.Lascoux and M. P. Schutzenberger, *Schubert polynomials and the Littlewood-Richardson Rule*, Let. Math. Phys. **10** (1985) 111-124.

[8] I. G. Macdonald, *Notes on Schubert Polynomials*, Publications du LACIM **6**, Universite du Quebec a Montreal (1991)bb

[9] L. Manivel. *Fonctions symtriques, polynmes de Schubert et lieux de dgnrescence.* Cours Spcialiss, 3. Socit Mathmatique de France, Paris, 1998.

[10] Alexander Postnikov, *On a Quantum Version of Pieri's Formula*, preprint March 23, 1997.

[11] F. Sottile, *Pieri's formula for flag manifolds and Schubert polynomials*, Annales de l'Institut Fourier **46** (1996), 89-110.

[12] Fulton, W., *Young tableaux : with applications to representation theory and geometry*, Cambridge University Press, 1997.

[13] R. Winkel. *A combinatorial bijection between Standard Young Tableaux and reduced words of Grassmannian permutations.* Sem. Loth. Comb. B36h (1996).